2.1 - Intro to Data Abstraction

Goal is to minimize *ambiguous code*.

data abstraction: obscuring how data is built so devs can focus on what it does instead

2.1.2 - Abstraction Barriers

Generally, any *functions or procedures* you make are usually abstractions of *tedious or repetitive tasks*.

Abstractions can be as simple as:

```
int addNums(int a, int b)
{
    return a + b;
}
```

...or simplify a complex task:

```
int catoi(char* input)
{
    int result = 0;

    // convert str to int
    int index;
    for (index = 0; index < cstrlen(input); index++)
    {
        result = result * 10 + input[index] - '0';
    }

    return result;
}
```

There are different levels of **data abstraction**, and the level you're working at is ambiguous and depends on how low-level your code is.

When creating your abstractions, you want to name them such that they do exactly what the name implies. For example, lets name a function that divides two numbers. We could name it `div`, but that what are we dividing? Integers? Fractions? We can get a general idea what that function *could* do, but we're not sure 100% what it does *exactly*. Instead `divideRationalNumbers` explains exactly what's going on.